Chapter 4

Regular Expressions

4.1 Some Definitions

Definition: If S and T are sets of strings of letters (whether they are finite or infinite sets), we define the product set of strings of letters to be

$$ST = \{w = w_1 w_2 : w_1 \in S,\ w_2 \in T\}$$

Example: If $S = \{a,\ aa,\ aaa\}$ and $T = \{b,\ bb\}$, then

$$ST = \{ab,\ abb,\ aab,\ aabb,\ aaab,\ aaabb\}$$

Example: If $S = \{a,\ ab,\ aba\}$ and $T = \{\Lambda,\ b,\ ba\}$, then

$$ST = \{a,\ ab,\ aba,\ abb,\ abba,\ abab,\ ababa\}$$

Example: If $S = \{\Lambda,\ a,\ aa\}$ and $T = \{\Lambda,\ bb,\ bbbb,\ bbbbbb,\ \ldots\}$, then

$$ST = \{\Lambda,\ a,\ aa,\ bb,\ abb,\ aabb,\ bbbb,\ abbbb,\ \ldots\}$$

Definition: Let s and t be strings. Then s is a *substring* of t if there exist strings u and v such that $t = usv$.

Example: Suppose $s = aba$ and $t = aababb$.
Then s is a substring of t since we can define $u = a$ and $v = bb$, and then $t = usv$.

Example: Suppose $s = abb$ and $t = aaabb$.
Then s is a substring of t since we can define $u = aa$ and $v = \Lambda$, and then $t = usv$.

Example: Suppose $s = bb$ and $t = aababa$.
Then s is *not* a substring of t.

Definition: Over the alphabet $\Sigma = \{a, b\}$, a string contains a *double letter* if it has either aa or bb as a substring.

Example: Over the alphabet $\Sigma = \{a, b\}$,

1. The string $abaabab$ contains a double letter.
2. The string bb contains a double letter.
3. The string aba does not contain a double letter.
4. The string $abbba$ contains two double letters.

4.2 Defining Languages Using Regular Expressions

Previously, we defined the languages:

- $L_1 = \{x^n \text{ for } n = 1,\ 2,\ 3,\ \ldots\}$
- $L_2 = \{x,\ xxx,\ xxxxx,\ \ldots\}$

But these are not very precise ways of defining languages.

- So we now want to be very precise about how we define languages, and we will do this using *regular expressions*.

- Languages that are associated with these regular expressions are called *regular languages* and are also said to be defined by a *finite representation.*

- Regular expressions are written in bold face letters and are a way of specifying the language.

- Recall that we previously saw that for sets S, T, we defined the operations

  - $S + T = \{w : w \in S \text{ or } w \in T\}$
  - $ST = \{w = w_1w_2 : w_1 \in S, w_2 \in T\}$
  - $S^* = S^0 + S^1 + S^2 + \cdots$
  - $S^+ = S^1 + S^2 + \cdots$

- We will precisely define what a regular expression is later. But for now, let's work with the following sketchy description of a regular expression.

- Loosely speaking, a regular expression is a way of specifying a language in which the only operations allowed are

  - union $(+)$,
  - concatenation (or product),
  - Kleene-$*$ closure,
  - superscript-$+$.

  The allowable symbols are parentheses, $\mathbf{\Lambda}$, and $\emptyset$, as well as each letter in Σ written in boldface. No other symbols are allowed in a regular expression. Also, a regular expression must only consist of a finite number of symbols.

- To introduce regular expressions, think of

  $$\mathbf{x} = \{x\};$$

  i.e., $\mathbf{x}$ represents the language (i.e., set) consisting of exactly one string, x. Also, think of

  $$\mathbf{a} = \{a\},$$
  $$\mathbf{b} = \{b\},$$

  so $\mathbf{a}$ is the language consisting of exactly one string a, and $\mathbf{b}$ is the language consisting of exactly one string b.

- Using this interpretation, we can interpret **ab** to mean

  $$\mathbf{ab} = \{a\}\{b\} = \{ab\}$$

  since the concatenation (or product) of the two languages $\{a\}$ and $\{b\}$ is the language $\{ab\}$.

- We can also interpret $\mathbf{a} + \mathbf{b}$ to mean

  $$\mathbf{a} + \mathbf{b} = \{a\} + \{b\} = \{a, b\}$$

- We can also interpret $\mathbf{a}^*$ to mean

  $$\mathbf{a}^* = \{a\}^* = \{\Lambda, a, aa, aaa, \ldots\}$$

- We can also interpret $\mathbf{a}^+$ to mean

  $$\mathbf{a}^+ = \{a\}^+ = \{a, aa, aaa, \ldots\}$$

- Also, we have

  $$(\mathbf{ab} + \mathbf{a})^*\mathbf{b} = (\{a\}\{b\} + \{a\})^*\{b\} = \{ab, a\}^*\{b\}$$

Example: Previously, we saw language

$$\begin{aligned} L_4 &= \{\Lambda,\ x,\ xx,\ xxx,\ \ldots\} \\ &= \{x\}^* \\ &= \text{language}(\mathbf{x}^*) \end{aligned}$$

Example: Language

$$\begin{aligned} L_1 &= \{x,\ xx,\ xxx,\ xxxx,\ \ldots\} \\ &= \text{language}(\mathbf{xx}^*) \\ &= \text{language}(\mathbf{x}^*\mathbf{x}) \\ &= \text{language}(\mathbf{x}^+) \\ &= \text{language}(\mathbf{x}^*\mathbf{xx}^*\mathbf{x}^*) \\ &= \text{language}(\mathbf{x}^*\mathbf{x}^+) \end{aligned}$$

Note that there are several different regular expressions associated with L_1.

Example: alphabet $\Sigma = \{a, b\}$
language L of all words of the form one a followed by some number (possibly zero) of b's.

$$L = \text{language}(\mathbf{ab}^*)$$

Example: alphabet $\Sigma = \{a, b\}$
language L of all words of the form some positive number of a's followed by exactly one b.

$$L = \text{language}(\mathbf{aa}^*\mathbf{b})$$

Example: alphabet $\Sigma = \{a, b\}$
language

$$L = \text{language}(\mathbf{ab}^*\mathbf{a}),$$

which is the set of all strings of a's and b's that have at least two letters, that begin and end with one a, and that have nothing but b's inside (if anything at all).

$$L = \{aa,\ aba,\ abba,\ abbba,\ \ldots\}$$

Example: alphabet $\Sigma = \{a, b\}$
The language L consisting of all possible words over the alphabet Σ has the following regular expression:

$$(\mathbf{a} + \mathbf{b})^*$$

Other regular expressions for L include $(\mathbf{a}^*\mathbf{b}^*)^*$ and $(\Lambda + \mathbf{a} + \mathbf{b})^*$.

Example: alphabet $\Sigma = \{x\}$
language L with an even number (possibly zero) of x's

$$\begin{aligned} L &= \{\Lambda,\ xx,\ xxxx,\ xxxxxx,\ \ldots\} \\ &= \text{language}((\mathbf{xx})^*) \end{aligned}$$

Example: alphabet $\Sigma = \{x\}$
language L with a positive even number of x's

$$\begin{aligned} L &= \{\ xx,\ xxxx,\ xxxxxx,\ \ldots\} \\ &= \text{language}(\mathbf{xx}(\mathbf{xx})^*) \\ &= \text{language}((\mathbf{xx})^+) \end{aligned}$$

Example: alphabet $\Sigma = \{x\}$
language L with an odd number of x's

$$\begin{aligned} L &= \{\, x,\ xxx,\ xxxxx,\ \ldots\} \\ &= \text{language}(\mathbf{x}(\mathbf{xx})^*) \\ &= \text{language}((\mathbf{xx})^*\mathbf{x}) \end{aligned}$$

Is $L = \text{language}(\mathbf{x}^*\mathbf{xx}^*)$?
No, since it includes the word $(xx)x(x)$.

Example: alphabet $\Sigma = \{a, b\}$
language L of all three-letter words starting with b

$$\begin{aligned} L &= \{baa,\ bab,\ bba,\ bbb\} \\ &= \text{language}(\mathbf{b}(\mathbf{a}+\mathbf{b})(\mathbf{a}+\mathbf{b})) \\ &= \text{language}(\mathbf{baa}+\mathbf{bab}+\mathbf{bba}+\mathbf{bbb}) \end{aligned}$$

Example: alphabet $\Sigma = \{a, b\}$
language L of all words starting with a and ending with b

$$\begin{aligned} L &= \{ab,\ aab,\ abb,\ aaab,\ aabb,\ abab,\ abbb,\ \ldots\} \\ &= \text{language}(\mathbf{a}(\mathbf{a}+\mathbf{b})^*\mathbf{b}) \end{aligned}$$

Example: alphabet $\Sigma = \{a, b\}$
language L of all words starting and ending with b

$$\begin{aligned} L &= \{b,\ bb,\ bab,\ bbb,\ baab,\ babb,\ bbab,\ bbbb,\ \ldots\} \\ &= \text{language}(\mathbf{b}+\mathbf{b}(\mathbf{a}+\mathbf{b})^*\mathbf{b}) \end{aligned}$$

Example: alphabet $\Sigma = \{a, b\}$
language L of all words with exactly two b's

$$L = \text{language}(\mathbf{a}^*\mathbf{ba}^*\mathbf{ba}^*)$$

Example: alphabet $\Sigma = \{a, b\}$
language L of all words with at least two b's

$$L = \text{language}((\mathbf{a}+\mathbf{b})^*\mathbf{b}(\mathbf{a}+\mathbf{b})^*\mathbf{b}(\mathbf{a}+\mathbf{b})^*)$$

Note that $bbaaba \in L$ since

$$bbaaba = (\Lambda)b(\Lambda)b(aaba) = (b)b(aa)b(a)$$

Example: alphabet $\Sigma = \{a, b\}$
language L of all words with at least two b's

$$L = \text{language}(\mathbf{a^*ba^*b(a+b)^*})$$

Note that $bbaaba \in L$ since $bbaaba = \Lambda\ b\ \Lambda\ b\ aaba$

Example: alphabet $\Sigma = \{a, b\}$
language L of all words with at least one a and at least one b

$$\begin{aligned} L &= \text{language}(\mathbf{(a+b)^*a(a+b)^*b(a+b)^*} + \mathbf{(a+b)^*b(a+b)^*a(a+b)^*}) \\ &= \text{language}(\mathbf{(a+b)^*a(a+b)^*b(a+b)^*} + \mathbf{bb^*aa^*}) \end{aligned}$$

where

- the first regular expression comes from separately considering the two cases:
  1. requiring an a before a b,
  2. requiring a b before an a.
- the second expression comes from the observation that the first term in the first expression only omits words that are of the form some b's followed by some a's.

Example: alphabet $\Sigma = \{a, b\}$
language L consists of Λ and all strings that are either all a's or b followed by a nonnegative number of a's

$$\begin{aligned} L &= \text{language}(\mathbf{a^*} + \mathbf{ba^*}) \\ &= \text{language}((\Lambda + \mathbf{b})\mathbf{a^*}) \end{aligned}$$

Theorem 5 *If L is a finite language, then L can be defined by a regular expression.*

Proof. To make a regular expression that defines the language L, turn all the words in L into boldface type and put pluses between them. ∎

Example: language

$$L = \{aba,\ abba,\ bbaab\}$$

Then a regular expression to define L is

$$\mathbf{aba} + \mathbf{abba} + \mathbf{bbaab}$$

4.3 The Language EVEN-EVEN

Example: Consider the regular expression

$$E = [\mathbf{aa} + \mathbf{bb} + (\mathbf{ab} + \mathbf{ba})(\mathbf{aa} + \mathbf{bb})^*(\mathbf{ab} + \mathbf{ba})]^*.$$

We now prove that the regular expression E generates the language EVEN-EVEN, which consists exactly of all strings that have an even number of a's and an even number of b's; i.e.,

$$\text{EVEN-EVEN} = \{\Lambda,\ aa,\ bb,\ aabb,\ abab,\ abba,\ baab,\ baba,\ bbaa,\ aaaabb,\ \ldots\}.$$

Proof.

- Let L_1 be the language generated by the regular expression E.
- Let L_2 be the language EVEN-EVEN.
- So we need to prove that $L_1 = L_2$, which we will do by showing that $L_1 \subset L_2$ and $L_2 \subset L_1$.
- First note that any word generated by E is made up of "syllables" of three types:

$$\begin{aligned}
\text{type}_1 &= \mathbf{aa} \\
\text{type}_2 &= \mathbf{bb} \\
\text{type}_3 &= (\mathbf{ab} + \mathbf{ba})(\mathbf{aa} + \mathbf{bb})^*(\mathbf{ab} + \mathbf{ba}) \\
E &= [\text{type}_1 + \text{type}_2 + \text{type}_3]^*
\end{aligned}$$

- We first show that $L_1 \subset L_2$:
  - Consider any string $w \in L_1$; i.e., w can be generated by the regular expression E.
  - We need to show that $w \in L_2$.
  - Note that since w can be generated by the regular expression E, the string w must be made up of syllables of type 1, 2, or 3.
  - Each of these types of syllables generate an even number of a's and an even number of b's.
    - type$_1$ syllable generates 2 a's and 0 b's.
    - type$_2$ syllable generates 0 a's and 2 b's.
    - type$_3$ syllable $(\mathbf{ab} + \mathbf{ba})(\mathbf{aa} + \mathbf{bb})^*(\mathbf{ab} + \mathbf{ba})$ generates
      - exactly 1 a and 1 b at the beginning,
      - exactly 1 a and 1 b at the end,
      - and generates either 2 a's or 2 b's at a time in the middle.
      - Thus, the type$_3$ syllable generates an even number of a's and an even number of b's.
  - Thus, the total string must have an even number of a's and an even number of b's.
  - Therefore, $w \in$ EVEN-EVEN, so we can conclude that $L_1 \subset L_2$.

- Now we want to show that $L_2 \subset L_1$; i.e., we want to show that any word with an even number of a's and an even number of b's can be generated by E.
  - Consider any string $w = w_1 w_2 w_3 \cdots w_n$ with an even number of a's and an even number of b's.
  - If $w = \Lambda$, then iterate the outer star of the regular expression E zero times to generate Λ.
  - Now assume that $w \neq \Lambda$.
  - Let $n = \text{length}(w)$.
  - Note that n is even since w consists solely of a's and b's and since the number of a's is even and the number of b's is even.
  - Thus, we can read in the string w two letters at a time from left to right.

- Use the following algorithm to generate $w = w_1 w_2 w_3 \cdots w_n$ using the regular expression E:
  1. Let $i = 1$.
  2. Do the following while $i \leq n$:
     (a) If $w_i = a$ and $w_{i+1} = a$, then iterate the outer star of E and use the type$_1$ syllable **aa**.
     (b) If $w_i = b$ and $w_{i+1} = b$, then iterate the outer star of E and use the type$_2$ syllable **bb**.
     (c) If ($w_i = a$ and $w_{i+1} = b$) or if ($w_i = b$ and $w_{i+1} = a$), then choose the type$_3$ syllable $(\mathbf{ab} + \mathbf{ba})(\mathbf{aa} + \mathbf{bb})^*(\mathbf{ab} + \mathbf{ba})$, and do the following:
       - If ($w_i = a$ and $w_{i+1} = b$), then choose **ab** in the first part of the type$_3$ syllable.
       - If ($w_i = b$ and $w_{i+1} = a$), then choose **ba** in the first part of the type$_3$ syllable.
       - Do the following while either ($w_{i+2} = a$ and $w_{i+3} = a$) or ($w_{i+2} = b$ and $w_{i+3} = b$):
         - Let $i = i + 2$.
         - If $w_i = a$ and $w_{i+1} = a$, then iterate the inner star of the type$_3$ syllable, and use **aa**.
         - If $w_i = b$ and $w_{i+1} = b$, then iterate the inner star of the type$_3$ syllable, and use **bb**.
       - Let $i = i + 2$.
       - If ($w_i = a$ and $w_{i+1} = b$), then choose **ab** in the last part of the type$_3$ syllable.
       - If ($w_i = b$ and $w_{i+1} = a$), then choose **ba** in the last part of the type$_3$ syllable.
       - Remarks:
         - We must eventually read in either ab or ba, which balances out the previous unbalanced pair. This completes a syllable of type$_3$.
         - If we never read in the second unbalanced pair, then either the number of a's is odd or the number of b's is odd, which is a contradiction.
     (d) Let $i = i + 2$.

- This algorithm shows how to use the regular expression E to generate any string in EVEN-EVEN; i.e., if $w \in$ EVEN-EVEN, then we can use the above algorithm to generate w using E.
- Thus, $L_2 \subset L_1$.

4.4 More Examples and Definitions

Example: $\mathbf{b}^*(\mathbf{abb}^*)^*(\Lambda + \mathbf{a})$ generates the language of all words without a double a.

Example: What is a regular expression for all valid variable names in C?

Definition: The set of regular expressions is defined by the following:

Rule 1 Every letter of Σ can be made into a regular expression by writing it in boldface; $\boldsymbol{\Lambda}$ and $\emptyset$ are regular expressions.

Rule 2 If $\mathbf{r}_1$ and $\mathbf{r}_2$ are regular expressions, then so are

1. $(\mathbf{r}_1)$
2. $\mathbf{r}_1\mathbf{r}_2$
3. $\mathbf{r}_1 + \mathbf{r}_2$
4. $\mathbf{r}_1^*$ and $\mathbf{r}_1^+$

Rule 3 Nothing else is a regular expression.

Definition: For a regular expression $\mathbf{r}$, let $L(\mathbf{r})$ denote the language generated by (or associated with) $\mathbf{r}$; i.e., $L(\mathbf{r})$ is the set of strings that can be generated by $\mathbf{r}$.

Definition: The following rules define the *language associated* with (or generated by) any regular expression:

Rule 1 (i) If $\ell \in \Sigma$, then $L(\boldsymbol{\ell}) = \{\ell\}$; i.e., the language associated with the regular expression that is just a single letter is that one-letter word alone.

(ii) $L(\boldsymbol{\Lambda}) = \{\Lambda\}$; i.e., the language associated with $\boldsymbol{\Lambda}$ is $\{\Lambda\}$, a one-word language.

(iii) $L(\boldsymbol{\emptyset}) = \emptyset$; i.e., the language associated with $\boldsymbol{\emptyset}$ is $\emptyset$, the language with no words.

Rule 2 If $\mathbf{r}_1$ is a regular expression associated with the language L_1 and $\mathbf{r}_2$ is a regular expression associated with the language L_2, then

(i) The regular expression $(\mathbf{r}_1)(\mathbf{r}_2)$ is associated with the language L_1 concatenated with L_2:

$$\text{language}(\mathbf{r}_1\mathbf{r}_2) = L_1L_2.$$

We define $\emptyset L_1 = L_1\emptyset = \emptyset$.

(ii) The regular expression $\mathbf{r}_1+\mathbf{r}_2$ is associated with the language formed by the union of the sets L_1 and L_2:

$$\text{language}(\mathbf{r}_1 + \mathbf{r}_2) = L_1 + L_2$$

(iii) The language associated with the regular expression $(\mathbf{r}_1)^*$ is L_1^*, the Kleene closure of the set L_1 as a set of words:

$$\text{language}(\mathbf{r}_1^*) = L_1^*$$

(iv) The language associated with the regular expression $(\mathbf{r}_1)^+$ is L_1^+:

$$\text{language}(\mathbf{r}_1^+) = L_1^+$$